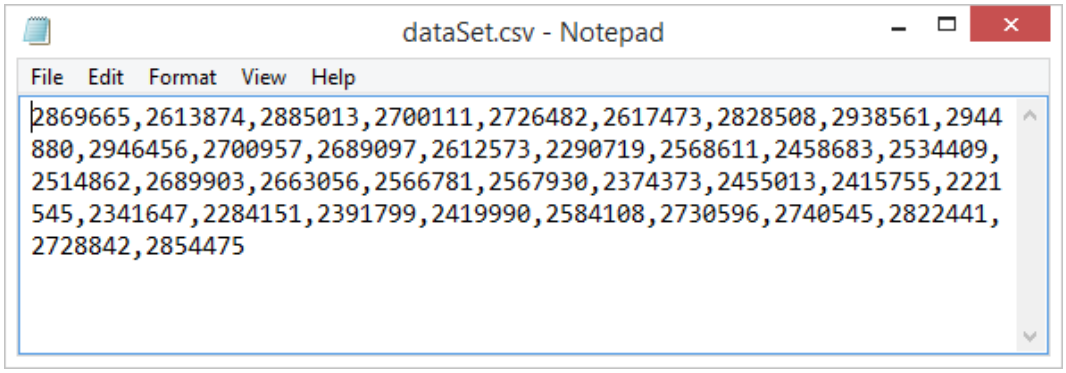
BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas tentang implementasi sistem prediksi tingkat produksi susu segar menggunakan metode *Support Vector Regression* yang dioptimasi dengan *Ant Colony Optimization*. Implementasi yang akan dibahas meliputi implementasi kode program dan implementasi *user interface*.

5.1 Implementasi Data

Berikut ini merupakan implementasi data *training* dan data *testing* yang dimasukkan ke dalam *file* dataset.csv



Gambar 5.1 Dataset

5.2 Implementasi Algoritma

Sistem prediksi tingkat produksi susu segar menggunakan metode *Support Vector Regression* yang dioptimasi dengan *Ant Colony Optimization* diimplementasikan menggunakan bahasa pemrograman JAVA dengan editor Netbeans. Implementasi algoritma meliputi implementasi kode program *Support Vector Regression* dan implementasi kode program *Ant Colony Optimization*.

5.2.1 Kode Program *Support Vector Regression*

Di bawah ini ditunjukkan kode program prediksi tingkat produksi susu segar menggunakan metode *Support Vector Regression*.

5.2.1.1 Perhitungan Jarak Data *Training*

Kode program ini merupakan fungsi yang digunakan untuk menghitung jarak antar data *training*. Hasil perhitungan jarak antar data *training* ini digunakan dalam perhitungan fungsi kernel RBF *training*.

```
1  public double[][] hitungJarakDataTraining(double[][] dt, int
2  fitur, int pjgDt) {
3          double[][] hasil = new double[pjgDt][pjgDt];
4          double[][] jarak = new double[pjgDt][pjgDt];
5          int pangkat = 2;
6          double tempJarak, hasill;
7          for (int i = 0; i < pjgDt; i++) {
8              for (int j = 0; j < pjgDt; j++) {
```

```
9                   tempJarak = 0.0;
0                   hasill = 0.0;
10                  for (int k = 0; k < fitur; k++) {
11                      tempJarak    =    Math.pow((dt[i][k]    -
12  dt[j][k]), pangkat);
13                      hasill += tempJarak;
14                      jarak[i][j] = hasill;
15                      hasil[i][j] = jarak[i][j];
16                  }
17              }
18          }
19          return hasil;
20      }
```

Kode Program 5.1 Perhitungan Jarak Data *Training*

Keterangan Kode Program 5.1 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3-4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5 merupakan inisialisasi variabel yang digunakan dalam *method*
4. Baris 6 merupakan deklarasi variabel yang digunakan dalam *method*
5. Baris 7-18 merupakan perulangan yang di dalamnya terdapat perhitungan jarak data training
6. Baris 19 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.2 Perhitungan Jarak Data *Testing*

Kode program ini merupakan fungsi yang digunakan untuk menghitung jarak antara data *testing* dan data *training*. Hasil perhitungan jarak antara data *testing* dan data *training* ini digunakan dalam perhitungan fungsi kernel RBF *testing*.

```
1   public      double[][]      hitungJarakDataTesting(double[][]
2   dtTraining, double[][] dtTesting, int fitur, int pjgDt, int
3   pjgTrain) {
4           double[][] hasil = new double[pjgDt][pjgTrain];
5           double[][] jarak = new double[pjgDt][pjgTrain];
6           int pangkat = 2;
7           double tempJarak, hasill;
8           for (int i = 0; i < pjgDt; i++) {
9               for (int j = 0; j < pjgTrain; j++) {
0                   tempJarak = 0.0;
10                  hasill = 0.0;
11                  for (int k = 0; k < fitur; k++) {
12                      tempJarak  =  Math.pow((dtTesting[i][k]  -
13  dtTraining[j][k]), pangkat);
14                      hasill += tempJarak;
15                      jarak[i][j] = hasill;
16                      hasil[i][j] = jarak[i][j];
17                  }
18              }
19          }
20          return hasil;
21      }
```

Kode Program 5.2 Perhitungan Jarak Data *Testing*

Keterangan Kode Program 5.2 :

1. Baris 1-3 merupakan deklarasi nama *method* dengan lima parameter
2. Baris 4-5 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 6 merupakan inisialisasi variabel yang digunakan dalam *method*
4. Baris 7 merupakan deklarasi variabel yang digunakan dalam *method*
5. Baris 8-19 merupakan perulangan yang di dalamnya terdapat perhitungan jarak data training
6. Baris 20 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.3 Perhitungan Kernel RBF *Training*

Kode program ini merupakan fungsi yang digunakan untuk menghitung kernel RBF data *training*. Hasil perhitungan kernel ini digunakan dalam perhitungan matriks Rij *training*.

```
1  public   double[][]   kernelRBFTraining(double[][]   dtJarak,
2  double sigma, int pjgDt) {
3          double[][] hasil = new double[pjgDt][pjgDt];
4          for (int i = 0; i < pjgDt; i++) {
5              for (int j = 0; j < pjgDt; j++) {
6                  hasil[i][j] = Math.exp(-(dtJarak[i][j] / (2 *
7  Math.pow(sigma, 2))));
8              }
9          }
10         return hasil;
11     }
```

Kode Program 5.3 Perhitungan Kernel RBF *Training*

Keterangan Kode Program 5.3 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-9 merupakan perulangan yang di dalamnya terdapat perhitungan nilai kernel RBF *training*
4. Baris 10 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.4 Perhitungan Kernel RBF *Testing*

Kode program ini merupakan fungsi yang digunakan untuk menghitung kernel RBF data *testing*. Hasil perhitungan kernel ini digunakan dalam perhitungan matriks Rij *testing*.

```
1  public double[][] kernelRBFTesting(double[][] dtJarak, double
2  sigma, int pjgDt, int pjgTrain) {
3          double[][] hasil = new double[pjgDt][pjgTrain];
4          for (int i = 0; i < pjgDt; i++) {
5              for (int j = 0; j < pjgTrain; j++) {
6                  hasil[i][j] = Math.exp(-(dtJarak[i][j] / (2 *
7  Math.pow(sigma, 2))));
```

```
8               }
9           }
10          return hasil;
11      }
```

Kode Program 5.4 Perhitungan Kernel RBF *Testing*

Keterangan Kode Program 5.4 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan empat parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-9 merupakan perulangan yang di dalamnya terdapat perhitungan nilai kernel RBF *testing*
4. Baris 10 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.5 Perhitungan Matriks Rij *Training*

Kode program ini merupakan fungsi yang digunakan untuk menghitung matriks Rij data *training*. Hasil perhitungan matriks Rij ini digunakan dalam perhitungan fungsi regresi f(x) *training*.

```
1   public double[][] hitungMatriksRijTraining(double[][] kernel,
2   double lamda, int pjgDt) {
3           double[][] hasil = new double[pjgDt][pjgDt];
4           for (int i = 0; i < pjgDt; i++) {
5               for (int j = 0; j < pjgDt; j++) {
6                   hasil[i][j] = kernel[i][j] + Math.pow(lamda,
7   2);
8               }
9           }
10          return hasil;
11      }
```

Kode Program 5.5 Perhitungan Matriks Rij *Training*

Keterangan Kode Program 5.5 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-9 merupakan perulangan yang di dalamnya terdapat perhitungan nilai matriks rij *training*
4. Baris 10 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.6 Perhitungan Matriks Rij *Testing*

Kode program ini merupakan fungsi yang digunakan untuk menghitung matriks Rij data *testing*. Hasil perhitungan matriks Rij ini digunakan dalam perhitungan fungsi regresi f(x) *testing*.

```
1   public double[][] hitungMatriksRijTesting(double[][] kernel,
2   double lamda, int pjgDt, int pjgTrain) {
3           double[][] hasil = new double[pjgDt][pjgTrain];
4           for (int i = 0; i < pjgDt; i++) {
5               for (int j = 0; j < pjgTrain; j++) {
6                hasil[i][j] = kernel[i][j] + Math.pow(lamda, 2);
```

```
7                 }
8             }
9             return hasil;
10        }
```

Kode Program 5.6 Perhitungan Matriks Rij *Testing*

Keterangan Kode Program 5.6 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan empat parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-8 merupakan perulangan yang di dalamnya terdapat perhitungan nilai matriks rij *testing*
4. Baris 9 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.7 Perhitungan Nilai E

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai E.

```
1   public double[] hitungNilaiE(double[][] dt, double[] alfa,
2   double[] alfa_bintang, double[][] rij, int kolom, int pjgDt)
3   {
4           double[] hasil = new double[pjgDt];
5           double E, totalE;
6           for (int i = 0; i < pjgDt; i++) {
7               E = 0.0;
8               totalE = 0.0;
9               for (int j = 0; j < pjgDt; j++) {
0                   E = (alfa_bintang[j] - alfa[j]) * rij[i][j];
10                  totalE += E;
11              }
12              hasil[i] = dt[i][kolom - 1] - totalE;
13          }
14          return hasil;
15      }
```

Kode Program 5.7 Perhitungan Nilai E

Keterangan Kode Program 5.7 :

1. Baris 1-3 merupakan deklarasi nama *method* dengan enam parameter
2. Baris 4-5 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 6-13 merupakan perulangan yang di dalamnya terdapat perhitungan nilai E
4. Baris 14 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.8 Perhitungan *Delta Alphai Star* ($\delta\alpha_i^*$)

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai *delta alphai star* ($\delta\alpha_i^*$).

```
1   public double[] hitungDeltaBintang(double gamma, double[] E,
2   double epsilon, double[] alfa_bintang, double c, int pjgnDt)
3   {
4           double[] hasil = new double[pjgnDt];
5           for (int i = 0; i < pjgnDt; i++) {
```

```
6               hasil[i] = Math.min(Math.max(gamma * (E[i] -
7   epsilon), -alfa_bintang[i]), c - alfa_bintang[i]);
8           }
9           return hasil;
10      }
11
```

Kode Program 5.8 Perhitungan *Delta Alphai Star* ($\delta\alpha_i^*$)

Keterangan Kode Program 5.8 :

1. Baris 1-3 merupakan deklarasi nama *method* dengan enam parameter
2. Baris 4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5-9 merupakan perulangan yang di dalamnya terdapat perhitungan nilai *delta alphai star*
4. Baris 10 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.9 Perhitungan *Delta Alphai* ($\delta\alpha_i$)

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai *delta alphai star* ($\delta\alpha_i$).

```
1   public double[] hitungDelta(double gamma, double[] E, double
2   epsilon, double[] alfa, double c, int pjgDt) {
3           double[] hasil = new double[pjgDt];
4           for (int i = 0; i < pjgDt; i++) {
5               hasil[i] = Math.min(Math.max(gamma * (-E[i] -
6   epsilon), -alfa[i]), c - alfa[i]);
7           }
8           return hasil;
9       }
```

Kode Program 5.9 Perhitungan *Delta Alphai* ($\delta\alpha_i$)

Keterangan Kode Program 5.9 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan enam parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-7 merupakan perulangan yang di dalamnya terdapat perhitungan nilai *delta alphai*
4. Baris 8 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.10 Perhitungan *Alphai Star* (α_i^*)

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai *alphai star* (α_i^*) setiap iterasi.

```
1   public double[] updateAlfaBintang(double[] alfa_bintang,
2   double[] deltaBintang, int pjgDt) {
3           double[] hasil = new double[pjgDt];
4           for (int i = 0; i < pjgDt; i++) {
5               alfa_bintang[i] = deltaBintang[i] +
6   alfa_bintang[i];
7               hasil[i] = alfa_bintang[i];
8           }
9           return hasil;
```

```
10      }
```

Kode Program 5.10 Perhitungan *Aphai Star* (α_i^*)

Keterangan Kode Program 5.10 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-8 merupakan perulangan yang di dalamnya terdapat perhitungan nilai *alphai star*
4. Baris 9 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.11 Perhitungan *Alphai* (α_i)

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai *alphai star* (α_i) setiap iterasi.

```
1  public double[] updateAlfa(double[] alfa, double[] delta, int
2  pjgDt) {
3          double[] hasil = new double[pjgDt];
4          for (int i = 0; i < pjgDt; i++) {
5              alfa[i] = delta[i] + alfa[i];
6              hasil[i] = alfa[i];
7          }
8          return hasil;
9      }
```

Kode Program 5.11 Perhitungan *Alphai* (α_i)

Keterangan Kode Program 5.11 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-7 merupakan perulangan yang di dalamnya terdapat perhitungan nilai *alphai*
4. Baris 8 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.12 Perhitungan F(x) *Training*

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai f(x) *training*.

```
1  public double[] hitungNilaiFXTraining(double[] alfa_bintang,
2  double[] alfa, double[][] rij, int pjgDt) {
3          double[] hasil = new double[pjgDt];
4          double FX, totalFX;
5          for (int i = 0; i < pjgDt; i++) {
6              FX = 0.0;
7              totalFX = 0.0;
8              for (int j = 0; j < pjgDt; j++) {
9                  FX = (alfa_bintang[j] - alfa[j]) * rij[i][j];
10                 totalFX += FX;
11                 hasil[i] = totalFX;
12             }
13         }
14         return hasil;
```

```
15      }
```

Kode Program 5.12 Perhitungan F(x) *Training*

Keterangan Kode Program 5.12 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan empat parameter
2. Baris 3-4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5-13 merupakan perulangan yang di dalamnya terdapat perhitungan nilai fx *training*
4. Baris 14 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.13 Perhitungan F(x) *Testing*

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai f(x) *testing*.

```
1   public double[] hitungNilaiFXTesting(double[] alfa_bintang,
2   double[] alfa, double[][] rij, int pjgDt, int pjgTrain) {
3           double[] hasil = new double[pjgDt];
4           double FX, totalFX;
5           for (int i = 0; i < pjgDt; i++) {
6               FX = 0.0;
7               totalFX = 0.0;
8               for (int j = 0; j < pjgTrain; j++) {
9                   FX = (alfa_bintang[j] - alfa[j]) * rij[i][j];
10                  totalFX += FX;
11                  hasil[i] = totalFX;
12              }
13          }
14          return hasil;
15      }
```

Kode Program 5.13 Perhitungan F(x) *Testing*

Keterangan Kode Program 5.13 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan lima parameter
2. Baris 3-4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5-13 merupakan perulangan yang di dalamnya terdapat perhitungan nilai fx *training*
4. Baris 14 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.14 Perhitungan Nilai MAPE

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai MAPE *training* dan MAPE *testing*.

```
1   public double MAPE(double[][] aktual, double[] prediksi, int
2   pjgData, int kolom) {
3           double mape, selisihData, totalSD = 0;
4           for (int i = 0; i < pjgData; i++) {
5               selisihData = Math.abs((aktual[i][kolom - 1] -
6   prediksi[i]) / aktual[i][kolom - 1]);
7               totalSD += selisihData;
8           }
```

```
9           mape = (totalSD / pjgData) * 100;
10          return mape;
11      }
```

Kode Program 5.14 Perhitungan Nilai MAPE

Keterangan Kode Program 5.14 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan empat parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-8 merupakan perulangan yang di dalamnya terdapat perhitungan nilai total selisih data
4. Baris 9 merupakan perhitungan nilai mape
5. Baris 10 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.15 Normalisasi Data

Kode program ini merupakan fungsi yang digunakan untuk normalisasi data *training* dan data *testing*.

```
1   public double[][] normalisasi(double[][] dt, double max,
2   double min) {
3           double[][]          hasil          =          new
4   double[dt.length][dt[0].length];
5           for (int i = 0; i < dt.length; i++) {
6               for (int j = 0; j < dt[0].length; j++) {
7                   hasil[i][j] = ((dt[i][j] - min) / (max -
8   min));
9               }
10          }
11          return hasil;
12      }
```

Kode Program 5.15 Normalisasi Data

Keterangan Kode Program 5.15 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3-4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5-10 merupakan perulangan yang di dalamnya terdapat perhitungan normalisasi data
4. Baris 11 merupakan fungsi pengembalian nilai variabel hasil

5.2.1.16 Denormalisasi Data

Kode program ini merupakan fungsi yang digunakan untuk denormalisasi data *training* dan data *testing*.

```
1   public double[] denormalisasi(double[] dt, double max, double
2   min) {
3           double[] hasil = new double[dt.length];
4           for (int i = 0; i < dt.length; i++) {
5               hasil[i] = ((dt[i] * (max - min)) + min);
6           }
7           return hasil;
```

```
8       }
```

Kode Program 5.16 Denormalisasi Data

Keterangan Kode Program 5.16 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 4-6 merupakan perulangan yang di dalamnya terdapat perhitungan denormalisasi data
4. Baris 7 merupakan fungsi pengembalian nilai variabel hasil

5.2.2 Kode Program *Ant Colony Optimization*

Di bawah ini ditunjukkan kode program optimasi parameter σ, c, ε yang dimiliki oleh *Support Vector Regression* menggunakan metode *Ant Colony Optimization*.

5.2.2.1 Perhitungan *ArgMax*

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai *argument max*. Fungsi ini digunakan untuk menentukan *next node* semut.

```
1   public int argMax(int r) {
2           int hasil = 0;
3           double max = pru(r, 0);
4           for (int i = 1; i < ordinate + 1; i++) {
5               if (max < pru(r, i)) {
6                   max = pru(r, i);
7                   hasil = i;
8               } else if (max == pru(r, i)) {
9                   int[] data = {hasil, i};
10                  hasil = data[(int) (Math.random() * 2)];
11              }
12          }
13          return hasil;
14      }
```

Kode Program 5.17 Perhitungan ArgMax

Keterangan Kode Program 5.17 :

1. Baris 1 merupakan deklarasi nama *method* dengan satu parameter
2. Baris 2-3 merupakan inisialisasi variabel yang digunakan dalam *method*
3. Baris 4-12 merupakan perulangan yang di dalamnya terdapat perhitungan nilai argmax
4. Baris 13 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.2 Perhitungan Probabilitas

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai probabilitas setiap feromon. Fungsi ini digunakan dalam perhitungan probabilitas kumulatif.

```
1       public double[] prs(int r) {
2           double[] prs = new double[ordinate + 1];
```

```
3           double[] hasil = new double[ordinate + 1];
4           double[] prsTemp = new double[ordinate + 1];
5           double totalPrs = 0;
6           //penyebut dalam prs
7           for (int i = 0; i <= ordinate; i++) {
8               prsTemp[i] = pru(r, i);
9               totalPrs += prsTemp[i];
10          }
11          //perhitungan probabilitas
12          for (int i = 0; i <= ordinate; i++) {
13              prs[i] = pru(r, i) / totalPrs;
14              hasil[i] = prs[i];
15          }
16          return hasil;
17      }
```

Kode Program 5.18 Perhitungan Probabilitas

Keterangan Kode Program 5.18 :

1. Baris 1 merupakan deklarasi nama *method* dengan satu parameter
2. Baris 2-4 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 5 merupakan inisialisasi variabel yang digunakan dalam *method*
4. Baris 7-15 merupakan perulangan yang di dalamnya terdapat perhitungan nilai probabilitas setiap *node*
5. Baris 16 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.3 Perhitungan Probabilitas Kumulatif

Kode program ini merupakan fungsi yang digunakan untuk menghitung nilai probabilitas kumulatif feromon. Fungsi ini digunakan dalam perhitungan *roulette wheel selection*.

```
1       public double[] prsCum(double[] prs) {
2           double[] prob = prs;
3           double[] hasil = new double[ordinate + 1];
4           double tempPrs = 0;
5           for (int i = 0; i <= ordinate; i++) {
6               tempPrs += prob[i];
7               hasil[i] = tempPrs;
8           }
9           return hasil;
10      }
```

Kode Program 5.19 Perhitungan Probabilitas Kumulatif

Keterangan Kode Program 5.19 :

1. Baris 1 merupakan deklarasi nama *method* dengan satu parameter
2. Baris 2-4 merupakan inisialisasi variabel yang digunakan dalam *method*
3. Baris 5-8 merupakan perulangan yang di dalamnya terdapat perhitungan nilai probabilitas kumulatif
4. Baris 9 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.4 *Roulette Wheel Selection*

Kode program ini merupakan fungsi yang digunakan untuk menghitung *roulette wheel selection*. Fungsi ini digunakan untuk menentukan *next node* semut.

```
1   public int rouletteWheelSelection(double[] prsCum) {
2           double[] probCum = prsCum;
3           int hasil = 0, indexSeleksi = 0;
4           double r;
5           r = (double) Math.random();
6           for (int i = 0; i < probCum.length; i++) {
7               if (r < probCum[i]) {
8                   indexSeleksi = i;
9                   hasil = indexSeleksi;
10                  break;
11              }
12          }
13          return hasil;
14      }
```

Kode Program 5.20 *Roulette Wheel Selection*

Keterangan Kode Program 5.20 :

1. Baris 1 merupakan deklarasi nama *method* dengan satu parameter
2. Baris 2-4 merupakan deklarasi dan inisialisasi variabel yang digunakan dalam *method*
3. Baris 5-12 merupakan perulangan yang di dalamnya terdapat penentuan *node* selanjutnya berdasarkan *roulette wheel selection*
4. Baris 13 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.5 Perhitungan *Update* Feromon Lokal

Kode program ini merupakan fungsi yang digunakan untuk memperbarui nilai feromon secara lokal ketika semut mengunjungi *next node*.

```
1   public double[][] updateFeromonLokal(int r, int u) {
2      double[][] data = feromon;
3      double[][]             hasil             =             new
4   double[data.length][data[0].length];
5      for (int i = 0; i < data.length; i++) {
6         for (int j = 0; j < data[0].length; j++) {
7             if ((i == r) && (j == u)) {
8                 data[i][j] = (((1 - rho) * (data[i][j])) + (rho
9   * feromonAwal));
0                 hasil[i][j] = data[i][j];
10            } else {
11                data[i][j] = data[i][j];
12                hasil[i][j] = data[i][j];
13                }
14            }
15          }
16          return hasil;
17      }
```

Kode Program 5.21 Perhitungan *Update* Feromon Lokal

Keterangan Kode Program 5.21 :

1. Baris 1 merupakan deklarasi nama *method* dengan dua parameter
2. Baris 2-4 merupakan deklarasi dan inisialisasi variabel yang digunakan dalam *method*
3. Baris 5-15 merupakan perulangan yang di dalamnya terdapat perhitungan untuk *update* feromon secara lokal dengan dua kondisi
4. Baris 16 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.6 Perhitungan *Update* Feromon Global

Kode program ini merupakan fungsi yang digunakan untuk memperbarui nilai feromon secara global ketika semua semut telah mengunjungi semua *node* setiap iterasi. Nilai feromon yang diperbarui yaitu feromon dengan jalur kunjungan yang memilliki MAPE paling kecil.

```
1  public double[][] updateFeromonGlobal(double[][] globalbest,
2  int bestIndexMAPE, int[][] kota) {
3     int[][] node = kota;
4     double[][] globalBest = globalbest;
5     double[][] hasil = feromon;
6     int bestIndexMape = bestIndexMAPE;
7     for (int i = 0; i < ordinate + 1; i++) {
8        for (int j = 0; j < ordinate + 1; j++) {
9            hasil[i][j] = ((1 - delta) * hasil[i][j]) + (delta
0  * 0);
10           for (int k = 1; k < node[0].length; k++) {
11               if (i == node[bestIndexMape][k - 1] && j ==
12 node[bestIndexMape][k]) {
13                   hasil[i][j] = ((1 - delta) * hasil[i][j]) +
14 (delta * (1 / globalBest[0][5])); //index mape = index ke 5
15                   }
16                }
17            }
18         }
19         return hasil;
20     }
```

Kode Program 5.22 *Update* Feromon Global

Keterangan Kode Program 5.22 :

1. Baris 1-2 merupakan deklarasi nama *method* dengan tiga parameter
2. Baris 3-6 merupakan deklarasi dan inisialisasi variabel yang digunakan dalam *method*
3. Baris 7-18 merupakan perulangan yang di dalamnya terdapat perhitungan untuk *update* feromon secara global
4. Baris 19 merupakan fungsi pengembalian nilai variabel hasil

5.2.2.7 Perhitungan Konversi Nilai *Sigma*, Kompleksitas, *Epsilon*, *CLR*, dan *Lambda*

Kode program ini merupakan fungsi yang digunakan untuk mengkonversi jalur kunjungan semut dengan MAPE terbaik setiap iterasi menjadi nilai parameter *sigma*, kompleksitas, *epsilon*, *cLR* dan *lambda* SVR.

```
1   public double[][] hasilOptimasi(int[][] kota, int m) {
2           double hasil[][] = new double[m][5];
3           double[] sigma = new double[m];
4           double[] c = new double[m];
5           double[] epsilon = new double[m];
6           double[] cLR = new double[m];
7           double[] lamda = new double[m];
8
9           for (int i = 0; i < sigma.length; i++) {
10              sigma[i] = ((kota[i][0] * 1000) + (kota[i][1] *
11  100) + (kota[i][2] * 10) + (kota[i][3]));
12              sigma[i] /= 1000;
13          }
14          for (int i = 0; i < c.length; i++) {
15              c[i] = ((kota[i][4] * 1000) + (kota[i][5] * 100)
16  + (kota[i][6] * 10) + (kota[i][7]));
17          }
18          for (int i = 0; i < epsilon.length; i++) {
19              epsilon[i] = ((kota[i][8] * 1000) + (kota[i][9] *
20  100) + (kota[i][10] * 10) + (kota[i][11]));
21              epsilon[i] /= 1000;
22          }
23          for (int i = 0; i < cLR.length; i++) {
24              cLR[i] = ((kota[i][12] * 1000) + (kota[i][13] *
25  100) + (kota[i][14] * 10) + (kota[i][15]));
26              cLR[i] /= 1000;
27          }
28          for (int i = 0; i < lamda.length; i++) {
29              lamda[i] = ((kota[i][16] * 1000) + (kota[i][17] *
30  100) + (kota[i][18] * 10) + (kota[i][19]));
31              lamda[i] /= 1000;
32          }
33          for (int i = 0; i < hasil.length; i++) {
34              int j = 0;
35              hasil[i][j] = sigma[i];
36              j++;
37              hasil[i][j] = c[i];
38              j++;
39              hasil[i][j] = epsilon[i];
40              j++;
41              hasil[i][j] = cLR[i];
42              j++;
43              hasil[i][j] = lamda[i];
44          }
45          return hasil;
46      }
```

Kode Program 5.23 Perhitungan Konversi Nilai Sigma*, Kompleksitas,* Epsilon*, CLR,* dan *Lambda*

Keterangan Kode Program 5.23 :

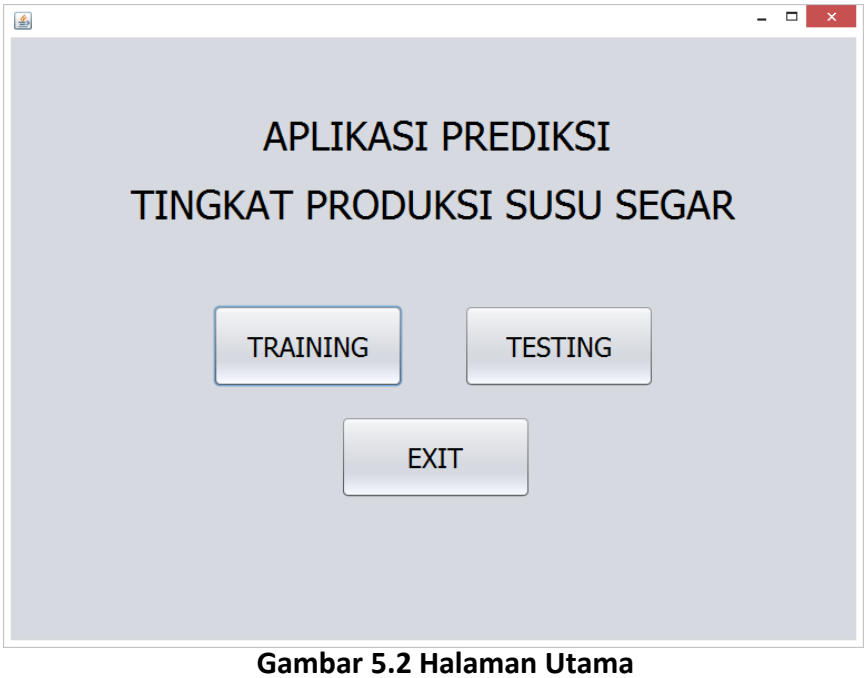
1. Baris 1 merupakan deklarasi nama *method* dengan dua parameter
2. Baris 2-7 merupakan deklarasi variabel yang digunakan dalam *method*
3. Baris 9-13 merupakan perulangan yang di dalamnya terdapat perhitungan untuk mengkonversi nilai *sigma*
4. Baris 14-17 merupakan perulangan yang di dalamnya terdapat perhitungan untuk mengkonversi nilai kompleksitas
5. Baris 18-22 merupakan perulangan yang di dalamnya terdapat perhitungan untuk mengkonversi nilai *epsilon*
6. Baris 23-27 merupakan perulangan yang di dalamnya terdapat perhitungan untuk mengkonversi nilai *cLR*
7. Baris 28-32 merupakan perulangan yang di dalamnya terdapat perhitungan untuk mengkonversi nilai *lambda*
8. Baris 35-44 merupakan perulangan untuk memasukkan kelima parameter SVR ke dalam satu matriks
9. Baris 45 merupakan fungsi pengembalian nilai variabel hasil

5.3 Implementasi *User Interface*

Implementasi *user interface* terdiri dari 4 halaman yaitu, halaman utama, halaman *training*, halaman *training result,* dan halaman *testing*.

5.3.1 Halaman Utama

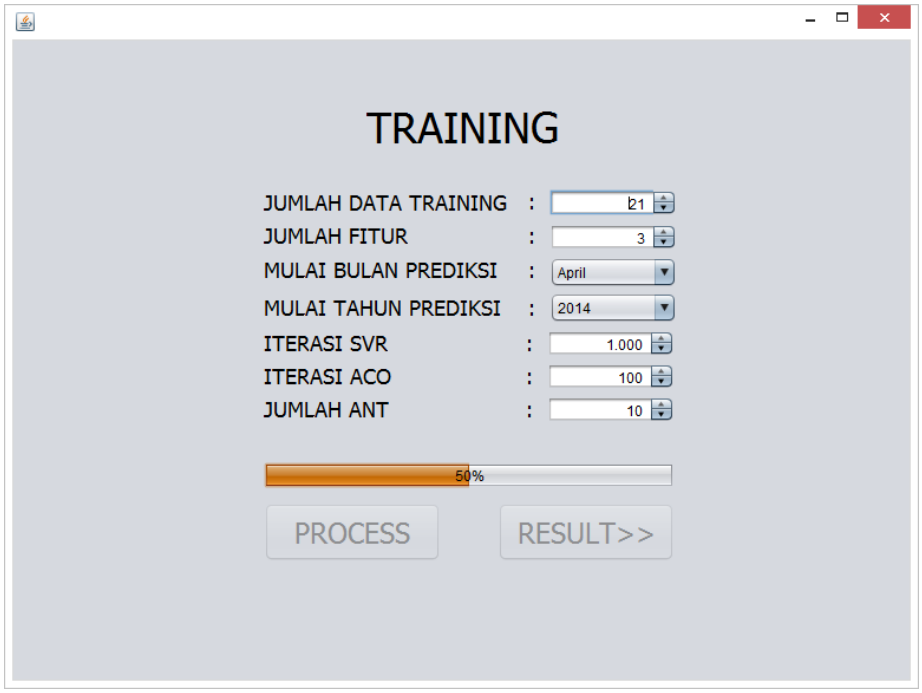
Di bawah ini merupakan implementasi halaman utama sistem.



Gambar 5.2 Halaman Utama

5.3.2 Halaman *Training*

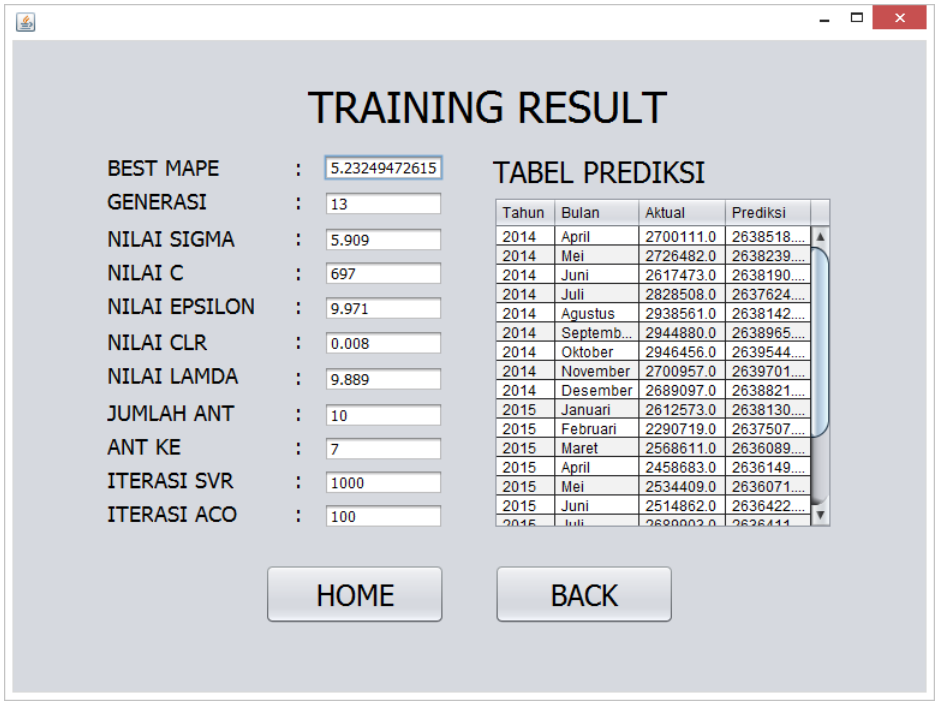
Di bawah ini merupakan implementasi halaman *training*.



Gambar 5.3 Halaman *Training*

5.3.3 Halaman *Training Result*

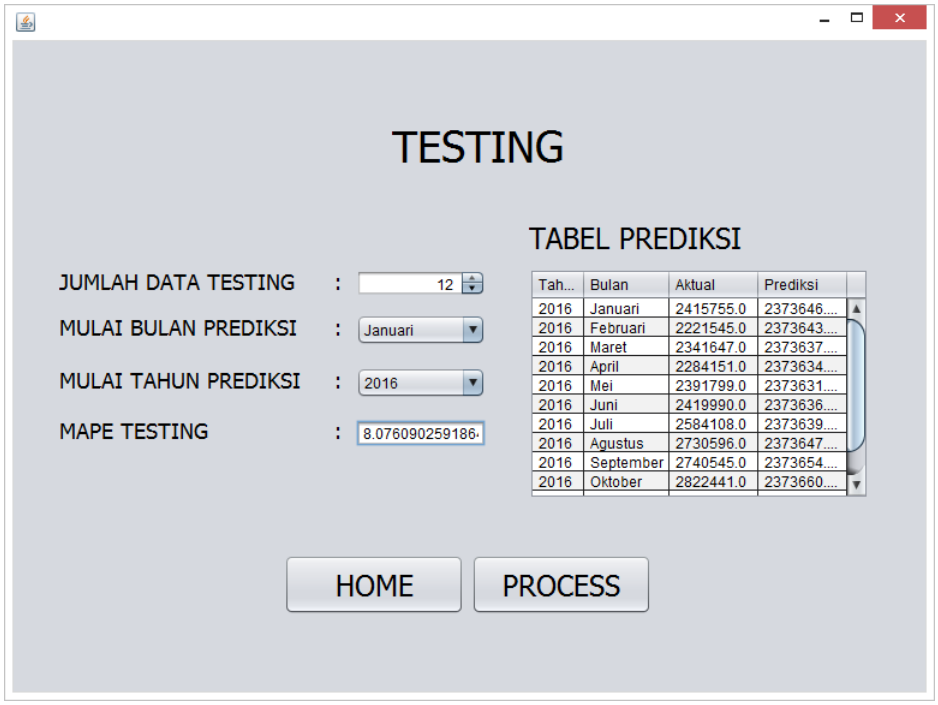
Di bawah ini merupakan implementasi halaman *training result*.



Gambar 5.4 Halaman *Training Result*

5.3.4 Halaman *Testing*

Di bawah ini merupakan implementasi halaman *testing*.



Gambar 5.5 Halaman *Testing*